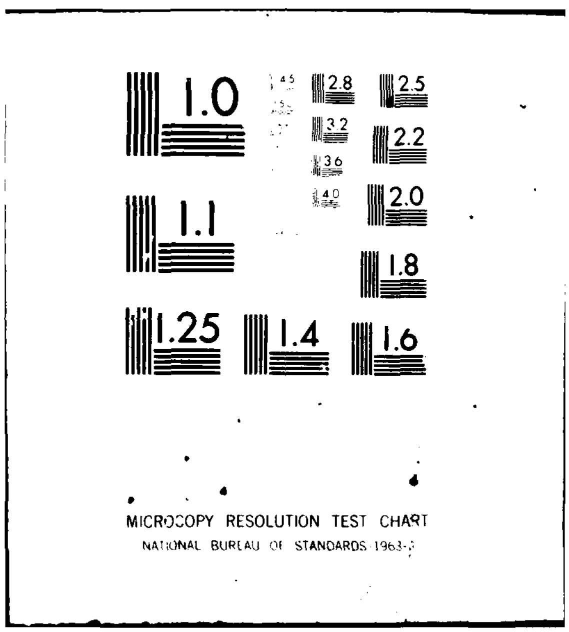
AD-A084 209 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/G 9/2
INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: A REVIEW OF THE GIFT--ETC(U)
MAR 80 N00014-79-C-0070
UNCLASSIFIED LMSC/D686137 NL

1 OF 1
AD
A084 209

END
DATE
FILMED
6-80
DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963

LEVEL II

12

ADA084209





This document has been approved for public release and sale; its distribution is unlimited.

LOCKHEED
MISSILES & SPACE COMPANY, INC. • SUNNYVALE, CALIFORNIA

12

# INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: A REVIEW OF THE GIFTS-STAGS UNION

Final Report
Contract N00014-79-C-0070 new

LMSC-D686137    31 March 1980





Prepared by:

G. M. Stanley
Applied Mechanics Laboratory
Department 52-33, Building 205
LOCKHEED PALO ALTO RESEARCH LABORATORY
3251 Hanover Street
Palo Alto, California 94304

## LIST OF ILLUSTRATIONS

# INTERACTIVE NONLINEAR STRUCTURAL ANALYSIS: A REVIEW OF THE GIFTS-STAGS UNION

## INTRODUCTION

Considering the three primary tasks were initially proposed [1], the software integration effort progressed successfully along the prescribed path. To summarize the objectives: (1) the GIFTS and STAGS finite element programs were to be data-base linked permitting GIFTS graphics-interactive pre- and post-processing of STAGS batch nonlinear structural analysis, (2) interactive software was to be introduced to facilitate convenient operation of the software assembly, and (3) the resulting integrated system was to be verified and documented. ~~To summarize~~ the corresponding results: (1) It is now possible to utilize GIFTS pre- and post-processing for a limited variety of STAGS analyses, (2) the software assembly is accessible to the user within a totally conversational environment in which both batch and interactive phases of analysis are prepared, managed and data-base monitored, and (3) current capabilities have been verified with respect to a modest selection of sample problems, and demonstrated to representatives of various U.S. Naval Laboratories. Before proceeding to finalize, it will be useful to examine the course and crux of this initial effort.

## APPROACH

Architecturally, the system evolved from the configuration depicted in Figure 1 into that depicted in Figure 2. (In these figures, blocks represent software elements and circles represent data elements.) The former configuration exhibited several major incompatibilities between GIFTS and STAGS: (1) a GIFTS counterpart for the STAGS1 (preprocessing + pre-analysis) program module did not exist, making a direct functional substitution impossible, (2) a STAGS counterpart for the GIFTS global data base (UDB) did not exist, making the prospect of data-base communication a formidable one and finally, (3) assuming that the previous two incompatibilities could be overcome through moderate software reconfiguration, the resulting combination of GIFTS interactive modules, STAGS batch modules and GIFTS ↔ STAGS interface modules would appear, and indeed be, intimidating for practical use.

Fortunately, a neat solution was available, as Figure 2 illustrates, with global data management serving as the key facilitating factor. The recently developed Lockheed global data-management utility, EZDAL [2], was used both to transform STAGS into an interfaceable program and to effect the needed automatic coordination. The current system thus evolved on the following sequence: (1) STAG1 was constructed from STAGS1 as a pre-analyzer with all conventional pre-processing functions removed; (2) EZDAL was implemented as the global data-manager for both STAG1 and STAG2 (essentially identical

to STAGS2), which involved the design of a formal STAGS global data base from which analysis could be initiated and into which results could be deposited and saved; (3) the GIFTS → STAGS transformer, TGS, was built to map the results of GIFTS preprocessing into the corresponding data required for STAG1 initiation, via a data-base-to-data-base transfer (see Figure 2); (4) similarly, the STAGS → GIFTS transformer, TSG, was built to map the results of a STAG2 solution into the corresponding data required for GIFTS post-processing; and (5) a control module, GIST, was built to preside over the software/data complex, interactively, with automatic monitoring and management of the STAGS global data-base provided via EZDAL.

To elucidate exactly what has been accomplished, and thus define the starting point for the continuation effort, each of the above steps will be examined in retrospect.

Step 1: Modularity Requirements

In step 1, STAGS1 was decomposed into two modules: a pure preprocessing module, STAG0, and a pre-analysis module, STAG1. The former module, which performs essentially the same functions as GIFTS' BULKM, EDITM, BULKLB and EDITLB (collectively), was quickly discarded upon completion of the GIFTS-STAGS interface. The latter module, or so-called pre-analyzer, performs a number of functions prerequisite to a staged analysis, but beyond the domain of model and load definition. For example, DOF number

assignment, element transformations, shape function and constitutive matrix evaluation at numerical integration points, and computational mass (diagonalized) and load vector assembly are typical STAG1 functions. As STAGS nonlinear analysis is based on a Lagrangian kinematic formulation, such functions need only be performed once, i.e., with respect to the reference configuration. Hence, STAG1 is a logically compact module and overcomes the first major incompatibility between GIFTS and STAGS.

Step 2: Global Data-Management Requirements

In step 2, EZDAL was implemented in all STAGS modules, thus establishing a formal global data base and achieving the modular independence initiated in step 1. This was a substantial effort requiring (1) the design of a data-base structure, (2) familiarity with the EZDAL data-management software utilities, and (3) the development of 'encapsulating' software for each STAGS module for transferring data to and from the global data base inconspicuously. The current organization of the STAGS global data base is illustrated in Figure 3. A single data module, i.e., file, contains all 'data sets' necessary to initiate and continue a STAGS analysis. As with the GIFTS global data base (or in GIFTS nomenclature, UDB) this data module is automatically declared as a permanent disk file by an executing program. Further, the GIFTS 'job name' has been adopted by the new STAGS program modules and the STAGS data module is therefore named jobnSTG, where jobn is the user's 4-character job name and STG is a fixed suffix. Comparing Figures 3 and 4, one sees that there are logical equivalences

between the data in the STAGS and GIFTS global data-bases, but physically, they differ in that GIFTS employs many data modules while STAGS employs only one containing many individual data-sets. This difference, which is practically transparent to the user, may be explained via the concept of 'local-global' data management.

First, we define 'global data base' as a collection of easily-accessible, labeled data existing on a permanent storage device(s) and sharing a common applications project or 'job'. Whereas GIFTS program modules employ a global data base (UDB) for all peripheral processing, STAGS program modules employ a temporary, local data base for computational purposes, resorting to the global data base only for finalized, archivable data. For manipulative efficiency, a local data base generally consists of multiple files; hence, since GIFTS does not distinguish between local and global, the UDB was conveniently designed in its current form. The preceding discussion should clarify what was meant by the 'encapsulating' software required to complete the STAGS implementation of EZDAL. This software, an element of which is called a Mover, is illustrated in Figure 5. The point is that Movers form the interface, or protective shell, between a program module and the outside world, i.e., the global data base. They simply transfer data from the global data base to the local data base at the beginning of program execution; and conversely, upon program termination (or periodically, e.g., following a

a major computation) transfer local data to the global data base. The EZDAL utilities for accessing the global data are embedded within these Movers.

Having developed an appropriate set of Movers, the STAGS implementation of EZDAL proceeded with minimal changes to the primary software. This, in fact, was one of the advantages of adopting the local-global approach. Once the second major incompatibility between GIFTS and STAGS had been removed, the next step was to utilize the STAGS global data base and its corresponding software utilities to construct the GIFTS-STAGS interface.

Step 3: Preprocessing Transformation

In step 3, the first Transformer was developed, TGS (Transformation from GIFTS to STAGS). This software module was the most time consuming to complete, as it led to the discovery of many minor incompatibilities between GIFTS and STAGS. For example, correspondence between certain GIFTS and STAGS finite elements did not exist, material and geometric property organization was inverted, many GIFTS modeling options, e.g., substructuring, had to be filtered from STAGS and the general controversy of global (cartesian) versus shell (surface) coordinates remained to be settled. Once it was concluded that, for expedience, the GIFTS-based definition of a STAGS model should be in terms of the STAGS 'general unit' rather than in terms of 'shell units', the resolution of incompatibilities went smoothly. This was,

of course, because the so-called 'general unit' of STAGS, which was devised to augment the definition of shell structures, employs global (cartesian) DOF and accepts a straightforward list of nodes and elements as input. The 'shell units', on the other hand, which were the basic building blocks for STAGS model definition, employ many special conventions concerning mesh generation, nodal transformations, inter-unit junctures and boundary conditions (to name a few). Unfortunately, this decision also precluded a number of desirable STAGS features, but the philosophy was to avoid trouble areas where possible until the totality of a basic system was completed. The TGS transformer is illustrated schematically in Figure 6 and organizationally in Figure 7a. As Figure 6 shows, a Transformer may be considered a 'long-distance' Mover, transferring data between two global data-bases (rather than between local and global). Consequently, TGS employs both GIFTS data-base utilities (G) and STAGS data-base utilities (S) to effect a model transformation. The only exception is when the module is exercised in 'coded' mode (Figure 6b), which will be required for multi-machine processing, whereupon only GIFTS utilities are employed by the Transformer.

Currently, TGS possesses the internal structure of Figure 7a, wherein each logically independent aspect of model definition is transformed from GIFTS to STAGS in a separate routine. The main routin  reatures a free-format command language so that the module may be operated in either batch or interactive mode.

Typically, TGS is exercised upon complete definition (or redefinition) of a GIFTS model, and enables STAGS analysis to follow by preparing that portion of the STAGS global data base expected by STAG1. The next task was to close the loop, requiring a STAGS → GIFTS transformer.

### Step 4: Post-Processing Transformation

In step 4, TSG was constructed on the same principles as TSG. The logical correspondences between STAGS solution data and GIFTS displayable results were more obvious than those sought in the previous Transformer, since the basic two quantities of concern to TSG are nodal vectors and element stresses. Thus, all STAGS vectoral quantities (displacements, velocities, eigenvectors, residuals, etc.) were transported to the GIFTS DNS (deflection) file and all stress-like quantities to the STR file (see Figure 7b). Due to the necessary reconfiguration of potentially large STAGS computational vectors into GIFTS logical form, STAGS Movers were incorporated in TSG to allow efficient local data-base manipulation. The limitations of TSG are reflected by the inability of GIFTS post-processing to properly accommodate all varieties of STAGS results.

### Step 5: Interactive Control

In step 5, the system was integrated by introducing the last software module, GIST. The GIST Control Module invokes all of the above apparatus via the computer operating system, while inter-

acting with the user in a conversational mode. Its basic role is to coordinate both the analysis and data corresponding to a given 'job' (Figure 8), using a set of functions which exhibit both interactive and automatic features. The basic functions shown in Figure 9 are pre-processing (PR command), analysis (AN command), and post-processing (PO command). Pre- and post-processing commands may invoke GIFTS interactive modules; whereas the analysis command leads to the launching of a conversationally prepared batch run stream, involving STAGS and the appropriate Transformer module(s). The activation of Transformers by GIST is one of the automatic features, and proceeds via status parameters set in the respective GIFTS and STAGS global data-bases. Regarding data-coordinating functions, GIST incorporates both GIFTS and STAGS global data-base utilities, enabling a direct, conversational link between user and data. Thus far, this has not been exploited except to obtain limited monitoring of the STAGS data base following batch analysis intervals. Retrospectively, much time was spent on the iterative refinement of the GIST module, due to the inherent difficulty of conversational programming and the struggle to isolate machine dependence. However, the resulting usability of the software system seems to justify the initial design effort.

## VERIFICATION

Finally, following many small-scale verifications, GIFTS-STAGS was applied to a practical problem (Figure 10) in which a multi-staged nonlinear static analysis was performed on a ring-stiffened cylindrical shell. The results matched those obtained in an earlier STAGS analysis up to a load level at which the difference in load hypotheses (i.e., 'live' pressure vs. stationary pressure) began to have noticeable effect.

## CONCLUSION

The major work of GIFTS-STAGS software integration has been accomplished and a powerful, but preliminary, product obtained. Upon concluding this initial effort, two observations are in order: (1) the collaboration with the GIFTS staff at the University of Arizona began very slowly and awkwardly, due to the initial ambiguity of the objectives; and (2) later, when the current architectural design had become solidified and partly implemented, the consulting relationship evolved into one of joint effort, i.e., the foundations for parallel development have now been established.

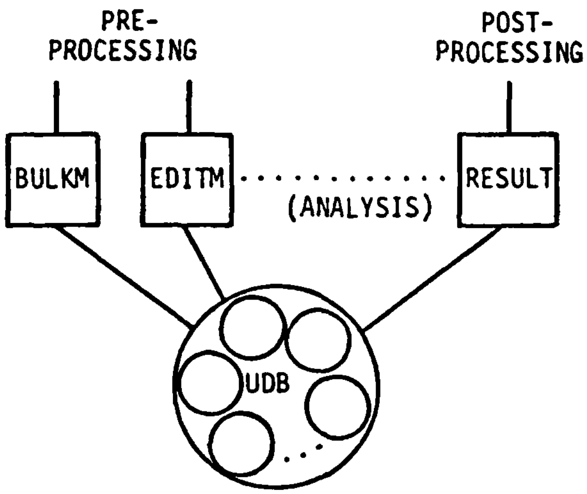
## REFERENCES


1. Stanley, G. M., "Interactive Nonlinear Structural Analysis: The GIFTS-STAGS Union", Proposal submitted to Office of Naval Research, August 1978.

2. Felippa, C. A., and Stearns, L. E., "The Global Data Base Manager, EZ-DAL", LMSC internal publication, January 1980.

FIGURE 1. THE PROBLEM: SOFTWARE INTEGRATION

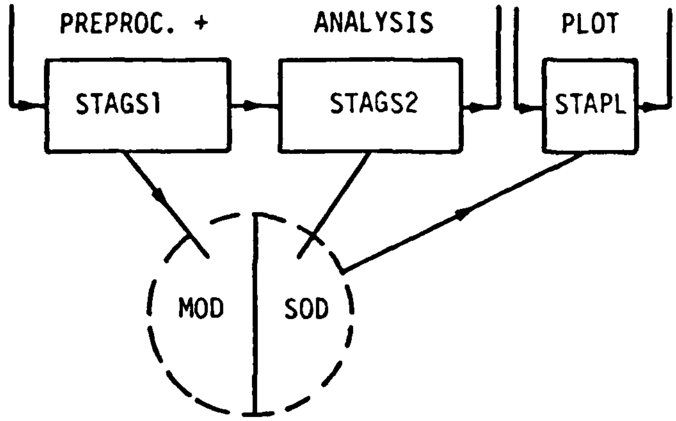
G I F T S




ATTRIBUTES:

- GRAPHICS/INTERACTIVE
- UNIFIED DATA-BASE (UDB)
  - LOCAL = GLOBAL
  - EFFICIENT, BUT COMPLEX
  - CONSISTENT
  - COMPLETE
  - EASY ACCESS
- USER DOCUMENTATION
  - USER'S MANUAL
  - THEORY MANUAL
  - HELP
- DEVELOPER DOCUMENTATION
  - SOFTWARE ORGANIZATION
  - SOFTWARE IMPLEMENTATION
  - THE UDB*
- A SEASONED PRE/POSTPROCESSING PACKAGE

S T A G S




ATTRIBUTES:

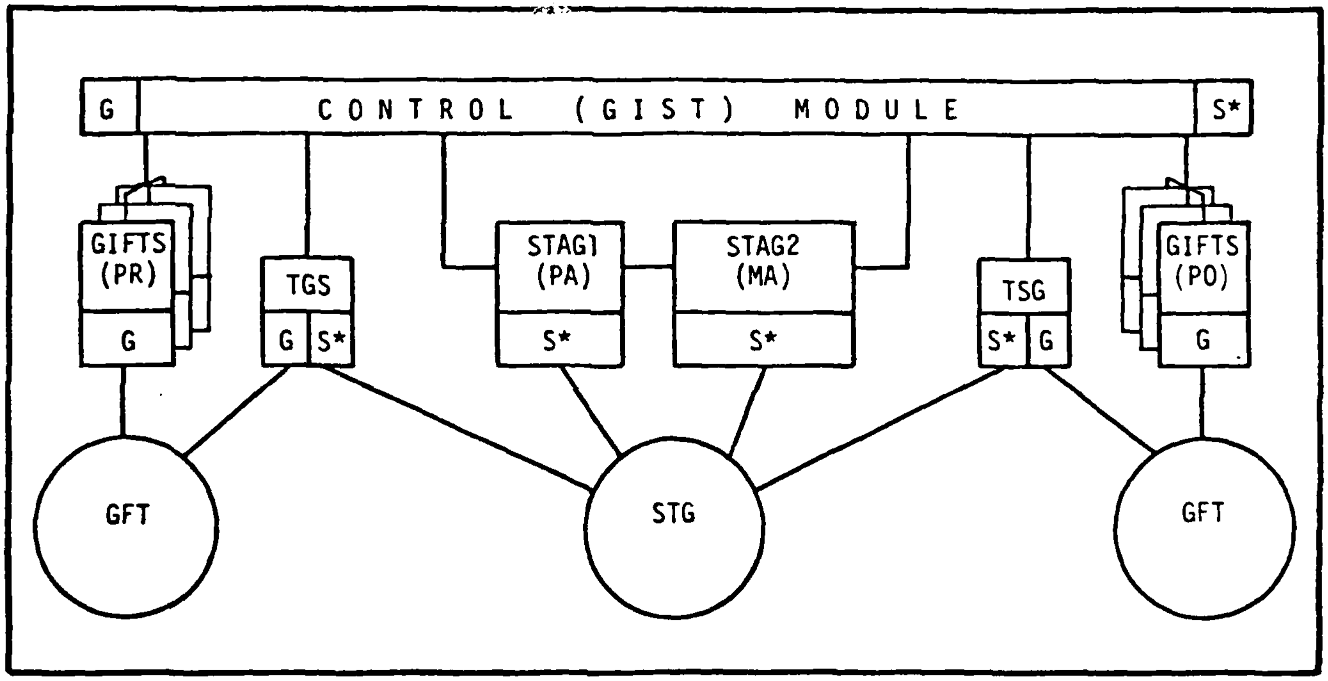
- BATCH
- EFFICIENT LOCAL DATA-MANAGEMENT
  - VMSYST/FM UTILITIES
- AD HOC GLOBAL DATA-MANAGEMENT
  - TRANSITORY
  - INCOMPLETE
  - DIFFICULT ACCESS
- USER DOCUMENTATION
  - USER'S MANUAL
  - THEORY MANUAL
  - BO, FRANK, GARY ...
- DEVELOPER DOCUMENTATION
  - "IN PREPARATION..."
- A SEASONED NONLINEAR STRUCTURAL ANALYZER

FIGURE 2. THE SOLUTION:

(1) Modularity

→ (2) Global Data-Management (EZDAL*)

(3) Control



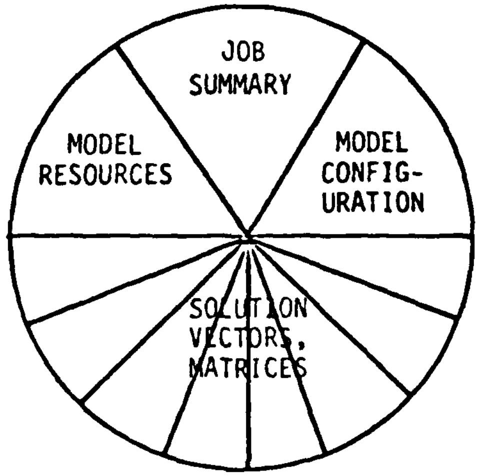

Attributes:

- Graphics-Interactive Pre/Post Processing
- Interactively "Launched" Batch Analysis
- \+ Universal Data-Management
- A Short User's Manual
- No Control Cards

- \- Some Casualties
- Not Yet "Street-Wise"

FIGURE 3. ORGANIZATION OF THE STAGS GLOBAL DATA-BASE

## Data-Module: "JOBN" STG




## Data-Sets

STAG."JOBN".SUMMARY

**Complex Record Structure**

| | | |
|---|---|---|
| STAG.RESO.MATL | . . . | Material Properties |
| STAG.RESO.FAB1 | . . . | 1-D Fabrications |
| STAG.RESO.FAB2 | . . . | 2-D Fabrications |
| STAG.STRU.UNIT.IU | . . . | Structural Units (nodes, elts, ..) |
| STAG.LOAD.SYST.IS | . . . | Loads, constraints, initial cond. |
| STAG.DIAD.O.ID | . . . | Directory to active DOF |

**Simple Record Structure**

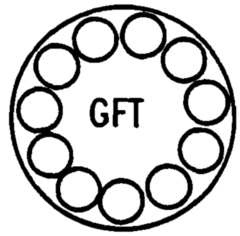
STAG.DISP."ISTEP" . . . Solution Vectors and Derived Quantities
STAG.VELO."ISTEP"

STAG.MODE."ISTEP"."MODE"

STAG.PLAS."ISTEP"
STAG.STRE."ISTEP"

FIGURE 4. ORGANIZATION OF THE GIFTS GLOBAL DATA-BASE

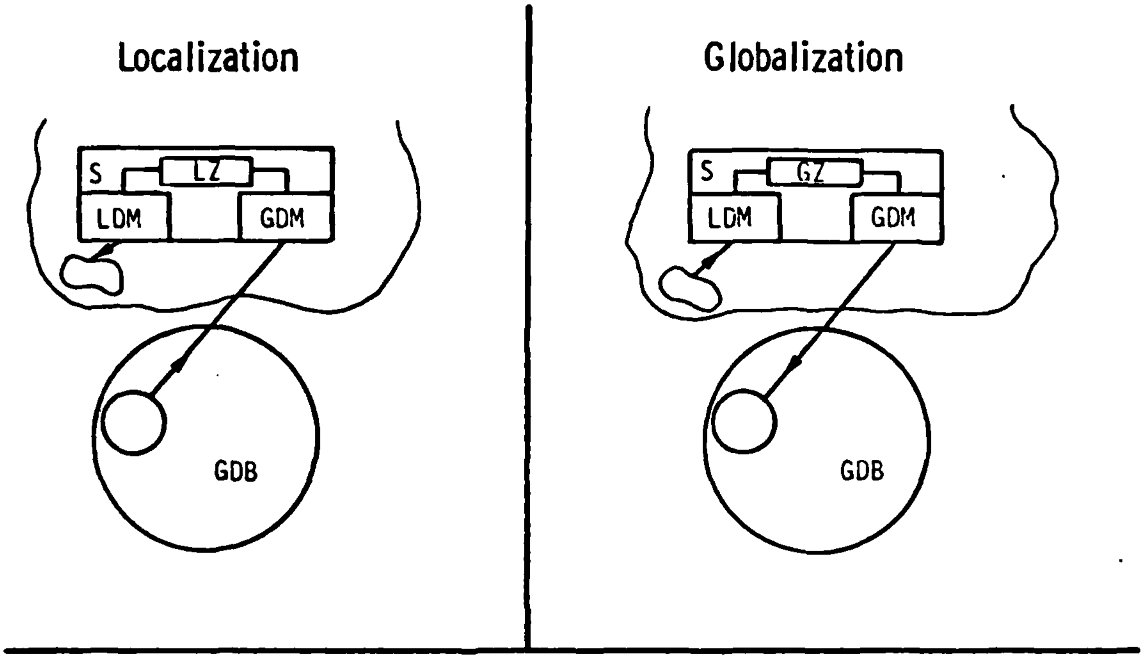
Data Modules




| | |
|---|---|
| PAR | Comprehensive list of problem parameters |
| FIL | Physical directory to all other data-modules |
| MAT | Material properties |
| THS | Geometric properties |
| PTS | Nodal characteristics |
| ELT | Element characteristics |
| GRD | Surface generation data |
| LIN | Line generation data |
| DNS | "Deflections" (Solution vectors displayable as deformed geometry) |
| STR | "Stresses" (Stresses, strains, stress resultants, residuals) |
| • | |
| • | |
| • | |

* Note: An effort to consolidate all of the above into a single data-module is in progress ("jobn" GFT).

FIGURE 5. THE STAGS MOVERS




Simple Movers (For Complex Operations)

- Entire section of LDB transferred as one "record"
- Header record describes size and "blocking" factor
- Examples: Vectors $\left\{\begin{matrix}\text{LZVEC}\\ \text{GZVEC}\end{matrix}\right\}$; Matrices $\left\{\begin{matrix}\text{LZMAT}\\ \text{GZMAT}\end{matrix}\right\}$

Complex Movers (For Simple Operations)

- Many logical records
- Many header records
- Examples: Model Resources $\left\{\begin{matrix}\text{LZMATL}\\ \text{GZMATL}\end{matrix}\right\}$, $\left\{\begin{matrix}\text{LZFAB2}\\ \text{GZFAB2}\end{matrix}\right\}$ - - -

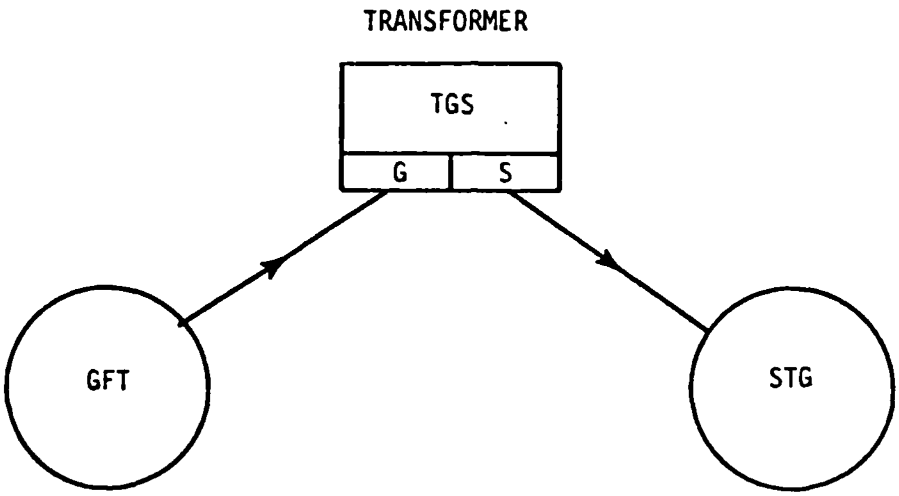
  Model Configuration $\left\{\begin{matrix}\text{LZUNIS}\\ \text{GZUNIS}\end{matrix}\right.$ (nodes, elements, etc.)$\left.\right\}$

General Purpose Movers

- Futuristic (Conventions?)

FIGURE 6. TRANSFORMERS (LONG-DISTANCE MOVERS)

(a) Direct (Standard Processing)




(b) Indirect (Distributed Processing)

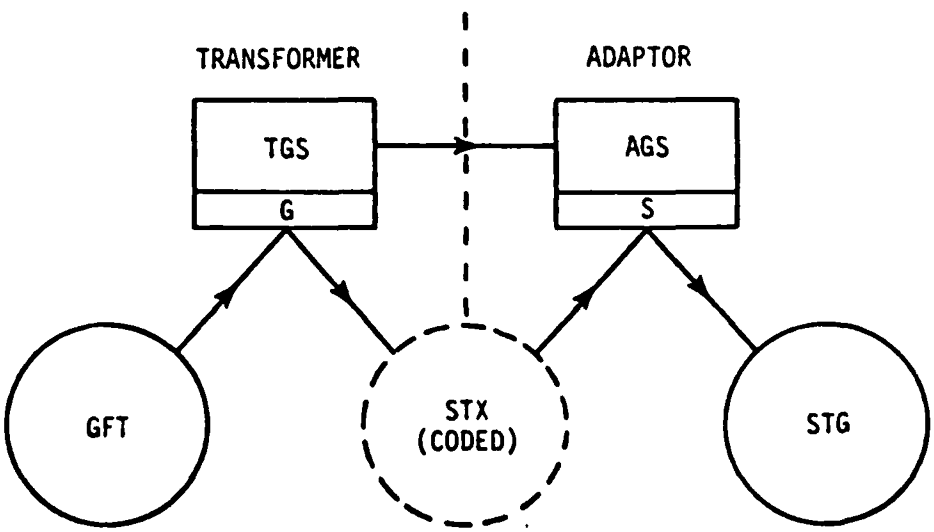
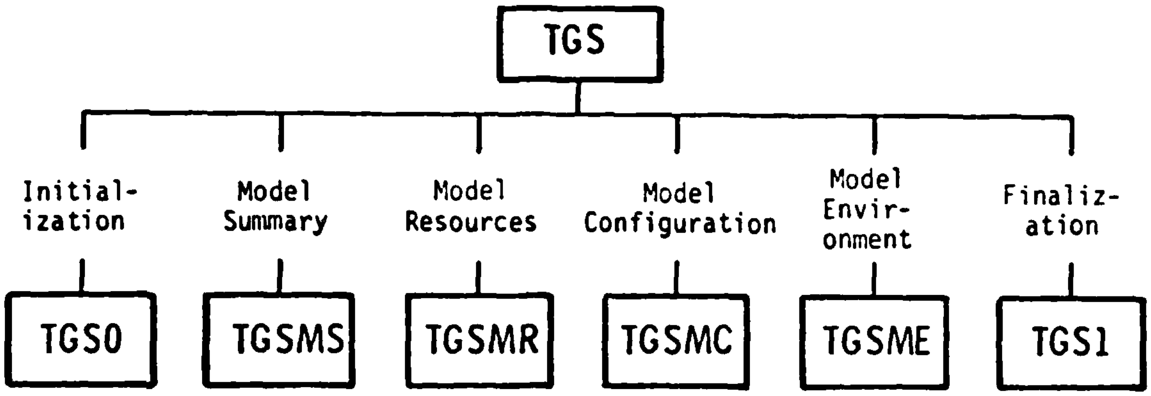


FIGURE 7. THE GIFTS ↔ STAGS TRANSFORMATION

(a) TGS: GIFTS → STAGS




(b) TSG: STAGS → GIFTS

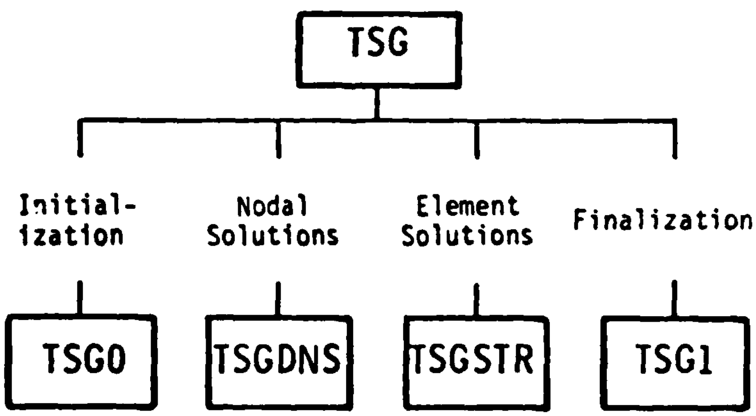
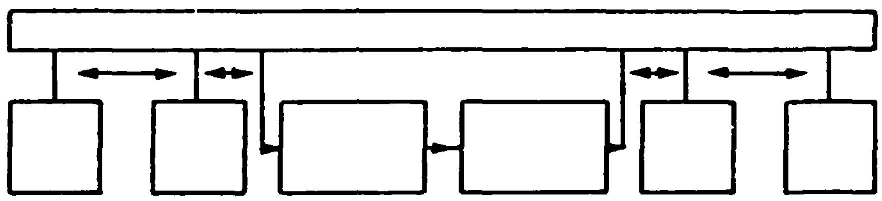


FIGURE 8. ROLE OF CONTROL

Analysis Coordination



Data Coordination

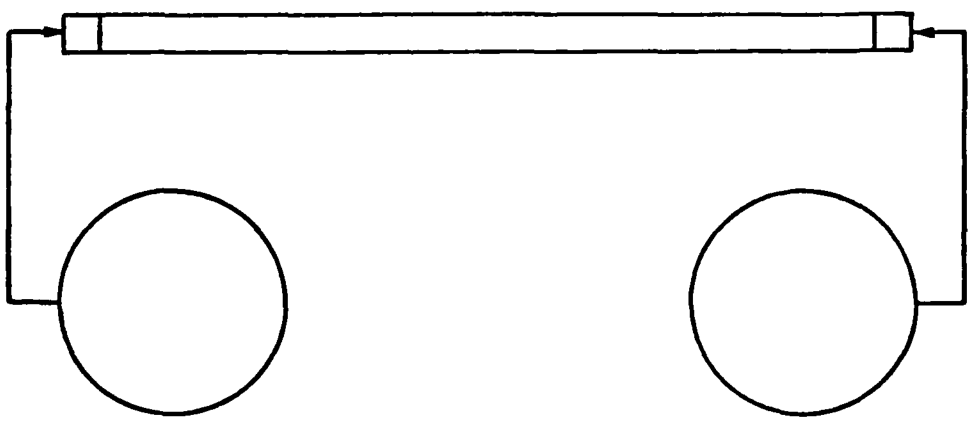
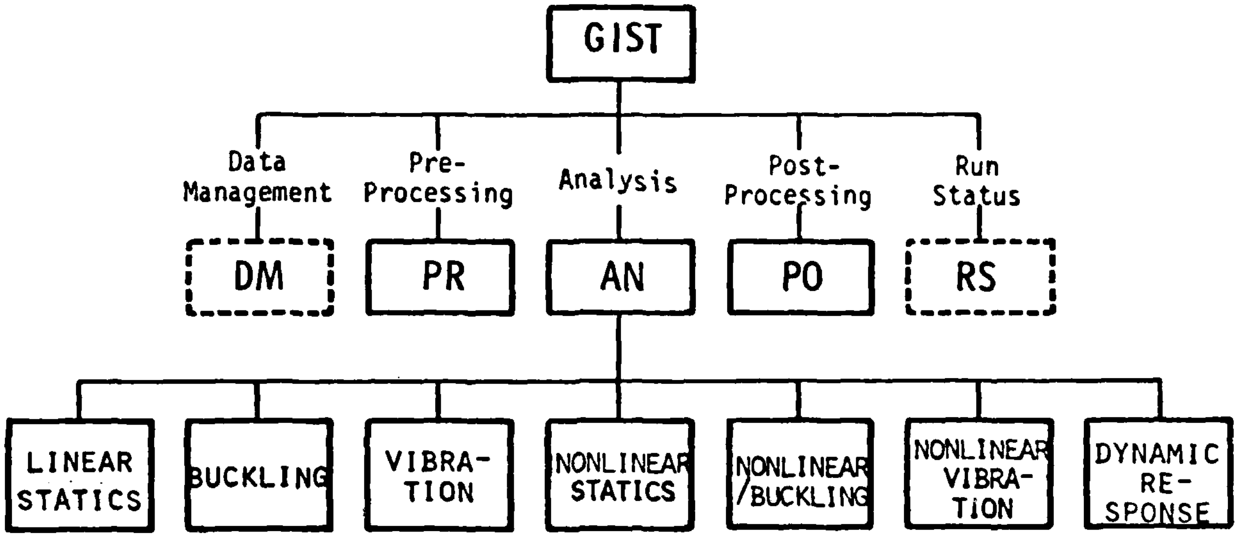
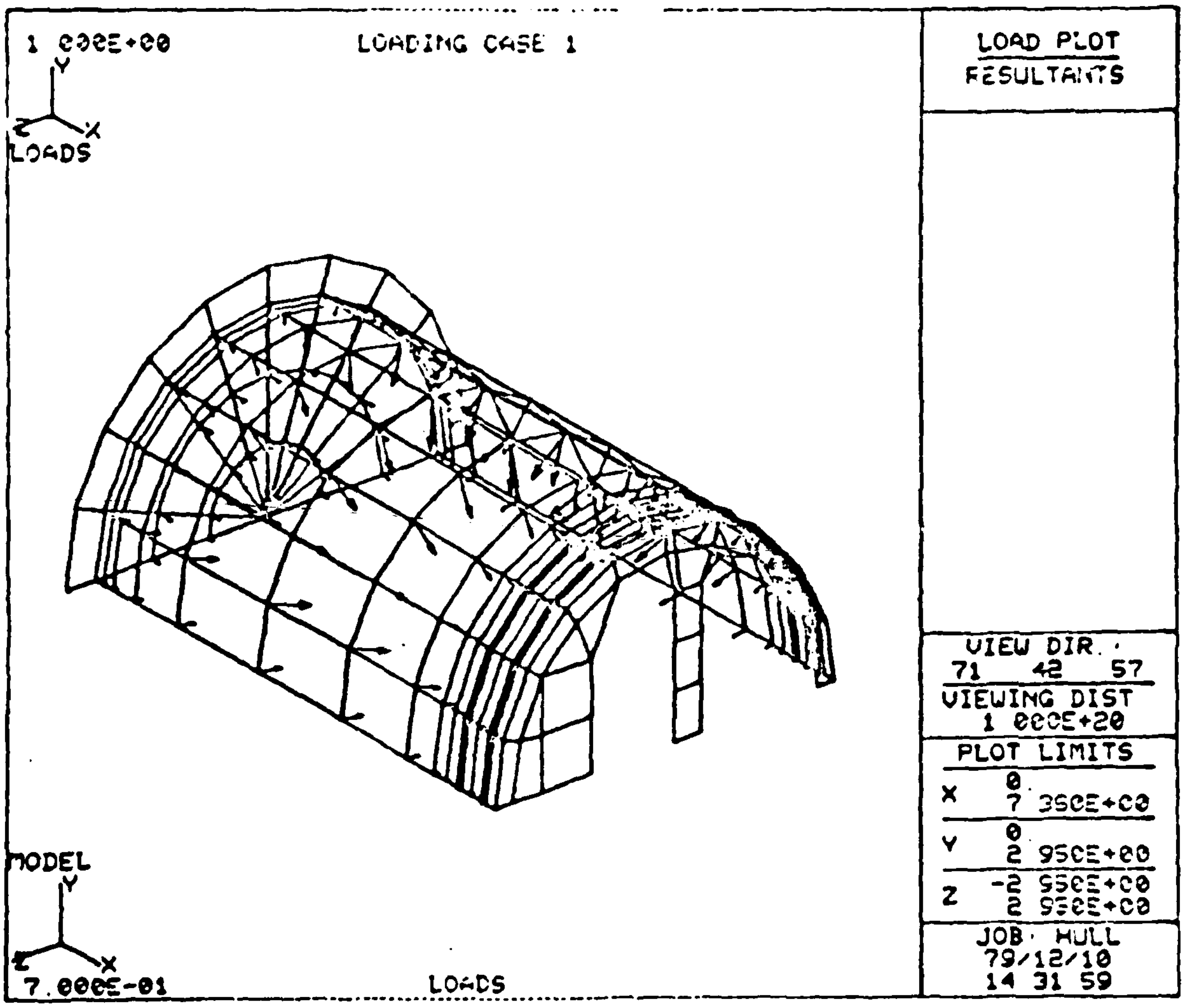


FIGURE 9. THE GIFTS -STAGS CONTROL MODULE: GIST




| Command | Function |
|---|---|
| PR: | • Access to GIFTS preprocessing modules<br>• Access to GIFTS → STAGS transformer<br>• Preprocessing status |
| AN: | • Preparation and "launch" of STAGS analysis<br>• Automatic activation of transformers<br>• Analysis status |
| PO: | • Access to GIFTS postprocessing modules<br>• Access to STAGS data-base<br>• Postprocessing status |
| DM: | • Command driven data-base management<br>• Data-base status |
| RS: | • Run status |
| HELP: | • Command summary<br>• Parameter descriptions |
| "Module": | • Direct transfer to interactive processors |

FIGURE 10. A DEMONSTRATION CASE



ILMED

8